Assessing the completeness evolution of DBpedia: A case study

Subhi Issa, Pierre-Henri Paris, and Fayçal Hamdi

CEDRIC - Conservatoire National des Arts et Métiers
292 rue saint martin, Paris, France

Abstract. RDF web datasets, thanks to their semantic richness, variety and fine granularity, are increasingly adopted by both researchers' and business communities. However, as anyone can publish data, this leads to sparse and heterogeneous data descriptions with undeniably an impact on quality. Consequently, there is an increasing effort dedicated to Web data quality improvement. We are interested in data quality and precisely in completeness quality evolution over time. The paper presents a set of experiments aiming to analyze the evolution of completeness quality values over several versions of DBpedia.

1 Introduction

While earlier datasets were relatively homogeneous and reasonably small, data on the Internet age, is more like a huge patchwork collected from various and numerous sources [10]. This data, even rich in content, is often incomplete and lacks metadata leading to unreliable analyses.

In this paper, we are interested in how the completeness of a dataset evolve. We believe that understanding this evolution could help define more suitable strategies for data sources integration, enrichment and maintenance. According to [8], the data completeness is about to know to which extent a dataset contains all of the necessary objects for a given task. As a consequence, a dataset may fit for a usage but not for another, and it is almost impossible to have an absolute measure of its completeness. By combining several types of measures about completeness of a given dataset, one can approach its real completeness value. In this paper, we do not aim at measuring an absolute completeness but rather measure a facet or an aspect of it. Our intuition is the following: things sharing the same class are described with some common properties. For a given class (in the sense of RDF type[1], or a DBpedia category in our precise case) there are properties that every instance of this class *should* have.

Example 1. Each person has a name, a birth place, etc. So, this should be reflected in the data, all the person instances *should* have all those properties. But not all people have a death place (at least for the moment), so this property may not be found in all the person instances.

[1] `https://www.w3.org/TR/rdf-schema/`

Hence, if a property p is sufficiently used among a set of instances of the same class T, we postulate that this property p is kind of mandatory (or may exist in the reality even if not present in the dataset) for all instances of this class T. Therefore, if p is missing for an instance of T, we may reasonably argue that this decreases the completeness of the dataset. Some data mining approaches may help to find those important set of properties.

We conducted an exploratory analysis of data completeness along several classes from DBpedia. We first defined a completeness assessment method that we applied in three versions of DBpedia. The remainder of this paper is organized as follows: Section 2 summarizes a related literature on the subject, then Section 3 exhibit a motivating example, while Section 4 details the completeness calculation approach. Section 5 presents and analyzes a set of experiments. Finally, Section 6 draws conclusions and future research directions.

2 Related works

Information quality attracted many research works for two decades. It is an interesting theoretical as well as practical domain and several projects proposed methodologies to help deal with quality assurance in traditional business information systems [1]. From the Web of Data perspective, the problem represents a new issue. Researchers pointed out the fact that making data accessible is not sufficient, especially when users are companies and governmental agencies, and when the target usage is business, research or countries' security. The credibility of data sources is thus bound to the quality of the content.

Several proposals could be classified using 4 categories from the TDQM[2] community [11] namely: *Intrinsic* (accuracy, reputation, believability and provenance), *Representational* (understandability, consistency and conciseness), *Accessibility* (accessibility and security), and *Contextual* (amount of data, relevance, completeness and timeliness). Intrinsic quality relies on internal characteristics of the data during evaluation. Most of the proposal concentrate on provenance data elicitation [5]. From an external point of view, representational quality is concerned with factors influencing users interpretations and practices such as understandability or data conciseness [12]. Concerning accessibility, a very important criterion in the context of web data, we could cite [7] where authors discussed common RDF publishers' errors that have a direct impact on data accessibility. Finally, contextual quality means that data could not be said "good" or "poor" without considering the context in which it is produced or used. The relevance dimension is considered from a ranking point of view in [2] and in the context of heterogeneous web data search in [6].

3 Motivating Example

We illustrate in this section the main idea behind our approach through an ex-

[2] Total Data Quality Management

ample that shows the issues and the difficulties encountered in the calculation of a dataset completeness. Let consider the set of scientists described in the well-known open linked dataset, DBpedia. We would like to know, when users querying this dataset about a scientist (or a subset of scientists), if the information provided for this scientist are complete (well described) or not. Taking for instance a subset of 100 scientists. The pseudo-code 1 returns, for each scientist, the couples $\langle property, value \rangle$.

Algorithm 1 Scientists Descriptions

```
String Query1 = "SELECT ?subject where{
                 ?subject rdf:type dbo:Scientist
                 } LIMIT 100"
Result S = ExecQuery(Query1)
for each subject ∈ S do
    String Query2 = "SELECT ?property ?value where{
                     subject ?property ?value}"
    Result R = ExecQuery(Query2)
    Descriptions.put(subject, < property, value >)
return Descriptions
```

To evaluate the completeness of this subset, a first intuition could consist of comparing the properties used in the description of each scientist with a reference scientist schema (ontology). For example, in DBpedia, the class *Scientist*[3] has a list of 4 properties (e.g. *doctoralAdvisor*), but these properties are not the only ones used in the description of a scientist (e.g. the *birthdate* property is not present in this list). Indeed, the class *Scientist* has a super class called *Person*. So, the description of a scientist may also take into account the properties of this class. Therefore, to obtain an exhaustive list of the whole properties used in the description of a scientist, we have to calculate the union of the set of properties of the class *Scientist* and all its ancestors. For our example, the reference scientist schema that we called *Scientist_Schema* could be calculated as follows:

$$Scientist\_Schema = \{Properties\ on\ Scientist\} \cup \{Properties\ on\ Person\} \cup \{Properties\ on\ Agent\} \cup \{Properties\ on\ Thing\}$$

$$\text{such that: } Scientist \sqsubseteq Person \sqsubseteq Agent \sqsubseteq Thing$$

Thus, the completeness of a scientist description (e.g. *Albert_Einstein*) will be the proportion of properties used in the description of this scientist to the total number of properties in *Scientist_Schema*. In the case of DBpedia, with a simple SPARQL query[4], we can obtain the size of *Scientist_Schema*, which

[3] `http://mappings.dbpedia.org/server/ontology/classes/`
[4] Performed on: `http://dbpedia.org/sparql`

is equal to 664 (A-Box properties). So, the completeness of the description of *Albert_Einstein* could be calculated as follows:

$$Comp(Albert\_Einstein) = \frac{|Properties\ on\ Albert\_Einstein|}{|Scientist\_Schema|} = \frac{21}{664} = 4,21\%$$

However, for example, the property *weapon* is in *Scientist_Schema*, but is not relevant for the *Albert_Einstein* instance.

We can finally conclude that, the completeness as calculated here, does not provide us with the relevant value regarding the real representation of scientists in the DBpedia dataset. Hence, to overcome this issue, we may have to explore those instances. We want to find which properties are used more often than others to describe instances of a given type. Based on data mining, the approach that we propose in this paper, deals with this issue by extracting, from a set of instances (of the same class), the set of the most representative properties and calculates completeness in respect to this set.

4 Completeness calculation: A Mining-based Approach

To assess the completeness of the different version of a LOD dataset (as DBpedia), we propose an approach that calculates the completeness of an input dataset by posing the problem as an itemset mining problem. In fact, the completeness at the data level assesses missing values [9]. This vision requires a schema (e.g. a set of properties) that needs to be inferred from the data source. However, it is not relevant to consider, for a subset of resources, the schema as the union of all properties used in their description as seen in Section 3. Indeed, this vision neglects the fact that missing values could express inapplicability. Our mining-based approach includes two steps:

1. **Properties mining**: Given a dataset $\mathcal{D}$, we first represent the properties, used for the description of the $\mathcal{D}$ instances, as a transaction vector. We then apply the well-known FP-growth algorithm [4] for mining frequent itemsets (we chose FP-growth for efficiency reasons. Any other itemset mining algorithm could, obviously, be used). Only a subset of these frequent itemsets, called "Maximal" [3], is captured. This choice is motivated by the fact that, on the one hand, we are interested in important properties for a given class that should appear often and, on the other hand, the number of frequent patterns could be exponential when the transaction vector is very large (see Section 4.1 for details).
2. **Completeness calculation**: Once the set of maximal frequent itemsets $\mathcal{MFP}$ is generated, we use the apparition frequency of items (properties) in $\mathcal{MFP}$, to give each of them a weight that reflects how important the set of properties is considered for the description of instances. Weights are then exploited to calculate the completeness of each transaction (regarding the

presence or absence of properties) and, hence, the completeness of the whole dataset.

In the following we give a detailed description of each step.

4.1 Properties mining

In this step, the objective is to find the properties sets that are the most shared by the subset of instances extracted from a dataset. Our assumption is that a property often used by several instances of a given type is more important than less often used properties for the same instances. This set will be then used to calculate a completeness value. More formally, let $\mathcal{D}(C, I_C, P)$ be a dataset, where C is the set of classes (e.g. rdf:type like *Actor*, *City*), I_C is the set of instances for categories in C (e.g., *Ben_Affleck* is an instance of the *Actor* class), and $P = \{p_1, p_2, ..., p_n\}$ is the set of properties (e.g. $residence(Person, Place)$). And let $\mathcal{I}'$ be a subset of data (instances) extracted from $\mathcal{D}$ with $\mathcal{I}' \subseteq I_C$. We first initialize $\mathcal{T} = \phi$, $\mathcal{MFP} = \phi$. For each $i \in \mathcal{I}'$ we generate a transaction t. Indeed, each instance i is related to values (either resources or literals) through a set of properties. Therefore, a transaction t_k of an instance i_k is a set of properties such that $t_k \subseteq P$. Transactions generated for all the instances of $\mathcal{I}'$ are then added to the $\mathcal{T}$ set.

Example 2. Taking table 1, let $\mathcal{I}'$ be a subset of instances such that: $\mathcal{I}' = \{$*The_Godfather, Goodfellas, True_Lies*$\}$. The set of transaction $\mathcal{T}$ would be:

$$\mathcal{T} = \{\{director, musicComposer\}, \{director, editing\}, \{director, editing, musicComposer\}\}$$

Table 1: A sample of DBpedia triples and their corresponding transactions

Subject	Predicate	Object
The_Godfather	director	Francis_Ford_Coppola
The_Godfather	musicComposer	Nino_Rota
Goodfellas	director	Martin_Scorsese
Goodfellas	editing	Thelma_Schoonmaker
True_Lies	director	James_Cameron
True_Lies	editing	Conrad_Buff_IV
True_Lies	musicComposer	Brad_Fiedel

Resource	Transaction
The_Godfather	{director, musicComposer}
Goodfellas	{director, editing}
True_Lies	{director, editing, musicComposer}

The objective is then to compute the set of frequent patterns $\mathcal{FP}$ from the transaction vector $\mathcal{T}$.

Definition 1. *(*Pattern*) Let $\mathcal{T}$ be a set of transactions. A pattern $\hat{P}$ is a sequence of properties shared by one or several transactions t in $\mathcal{T}$.*

For any pattern $\hat{P}$ (e.g. a set of properties), let $T(\hat{P}) = \{t \in \mathcal{T} \mid \hat{P} \subseteq E(t)\}$ be the corresponding set of transactions. $|T(\hat{P})|$ the *support* of $\hat{P}$ (e.g. the number of individuals having all properties of $\hat{P}$). A pattern $\hat{P}$ is frequent if $\frac{1}{|\mathcal{T}|}|T(\hat{P})| \geq \xi$, where ξ is a user-specified threshold.

Example 3. Taking table 1, let $\hat{P} = \{director, musicComposer\}$ and $\xi = 60\%$. $\hat{P}$ is frequent as its relative support (66.7%) is greater than ξ.

To find all the frequent patterns $\mathcal{FP}$, we used, as we motivated above, the FP-growth itemsets mining algorithm. However, according to the size of the transactions vector, the FP-growth algorithm could generate a very large $\mathcal{FP}$ set. Furthermore, we need only frequent enough patterns. In itemset mining, a concept, called "Maximal" frequent patterns, allow us to find those subsets of properties. Thus, to reduce $\mathcal{FP}$, we generate a subset containing only "Maximal" patterns.

Definition 2. *($\mathcal{MFP}$) Let $\hat{P}$ be a frequent pattern. $\hat{P}$ is maximal if none of its proper superset is frequent. We define the set of Maximal Frequent Patterns $\mathcal{MFP}$ as:*

$$\mathcal{MFP} = \{\hat{P} \in \mathcal{FP} \mid \forall \hat{P}' \supsetneq \hat{P} : \frac{|T(\hat{P}')|}{|\mathcal{T}|} < \xi\}$$

Example 4. Taking table 1, let $\xi = 60\%$ and the set of frequent patterns $\mathcal{FP} =$ {{*director*},{*musicComposer*},{*editing*}, {*director,musicComposer*} {*director,editing*}}. The $\mathcal{MFP}$ set would be: $\mathcal{MFP}$={{*director,musicComposer*},{*director, editing*}}

4.2 Completeness calculation

In this step, we carry out for each transaction, a comparison between its corresponding properties and each pattern of the $\mathcal{MFP}$ set (regarding the presence or the absence of the pattern). An average is, therefore, calculated to obtain the completeness of each transaction $t \in \mathcal{T}$. Finally, the completeness of the whole $t \in \mathcal{T}$ will be the average of all the completeness values calculated for each transaction.

Definition 3. *(*Completeness*) Let $\mathcal{I}'$ a subset of instances, $\mathcal{T}$ the set of transactions constructed from $\mathcal{I}'$, and $\mathcal{MFP}$ a set of maximal frequent pattern. The completeness of $\mathcal{I}'$ corresponds to the completeness of its transaction vector $\mathcal{T}$ obtained by calculating the average of the completeness of $\mathcal{T}$ regarding each pattern in $\mathcal{MFP}$. Therefore, we define the completeness $\mathcal{CP}$ of a subset of instances $\mathcal{I}'$ as follows:*

$$\mathcal{CP}(\mathcal{I}') = \frac{1}{|\mathcal{T}|} \sum_{k=1}^{|\mathcal{T}|} \sum_{j=1}^{|\mathcal{MFP}|} \frac{\delta(E(t_k), \hat{P}_j)}{|\mathcal{MFP}|} \tag{1}$$

$$\text{such that: } \hat{P}_j \in \mathcal{MFP}, \text{ and } \delta(E(t_k), \hat{P}_j) = \begin{cases} 1 \text{ if } \hat{P}_j \subset E(t_k) \\ 0 \text{ otherwise} \end{cases}$$

Example 5. Let $\xi = 60\%$. The completeness of the subset of instances in table 1 regarding $\mathcal{MFP} = \{\{director, musicComposer\}, \{director, editing\}\}$, would be:

$$\mathcal{CP}(\mathcal{I}') = (2 * (1/2) + (2/2))/3 = 0.67$$

This value corresponds to the completeness average value for the whole dataset regarding the inferred patterns in $\mathcal{MFP}$.

5 Experiments

The experiments were performed on the well-known real-world datasets, DBpedia, publicly available on the Linked Open Data (LOD). DBpedia, is a large knowledge base composed of structured information extracted collaboratively from Wikipedia. It describes currently about 14 million things.

For evaluating the completeness of different versions of DBpedia, we chose three relatively distant versions. The first one (v3.6) was generated in March/April 2013, the second one (v2015-04) in February/March 2015 and the third one (v2016-10) in October 2016. For each dataset we have chosen a couple of classes from different natures. We studied the completeness of resources that have as classes the following ones: $C = \{$*Film*, *Organisation*, *Scientist*, *PopulatedPlace*$\}$. For the properties used in the resources descriptions, we have chosen the English datasets "mapping-based properties", "instance types" and "labels". The number of triples (statements) of each class is given in Table 2.

Table 2: Number of resources/class

	Film	Organisation	PopulatedPlace	Scientist
v3.6(2013)	53,619	147,889	340,443	9,726
v2015-04	90,060	187,731	455,398	20,301
v2016-10	106,613	275,077	505,557	23,373

In the first step of our experiments, we constructed the set of corresponding transactions $\mathcal{T}$. A transaction vector is constituted of sequences of properties deduced from instances belonging to a single class (e.g. the set of *Film* in DBpedia). The set of transactions is then used as an input to generate frequent patterns and to compute the completeness. Experiments were run on a Dell XPS27 with an Intel Core i7-4770S processor and 16 GB of DDR3 RAM. The execution time of each experiment is about 2 minutes.

The evaluation methodology consists of calculating, regarding the same inferred schema (for us the same $\mathcal{MFP}$), the completeness of different versions. We chose, as a reference schema, for our experiments the one inferred from the older version. Thus, we can observe the completeness evolution over versions.

We performed a set of experiments on which completeness calculation was performed only on equivalent resources belonging to the three versions (intersection of triples subjects). Therefore we focus on how the completeness is impacted by updated (not inserted) data. Figure 1 shows the completeness results obtained for the chosen classes of DBpedia v3.6, v2015-04 and v2016-10 when varying the minimum support ξ. The completeness is calculated for the three versions regarding the same $\mathcal{MFP}$ inferred from the v3.6 version.

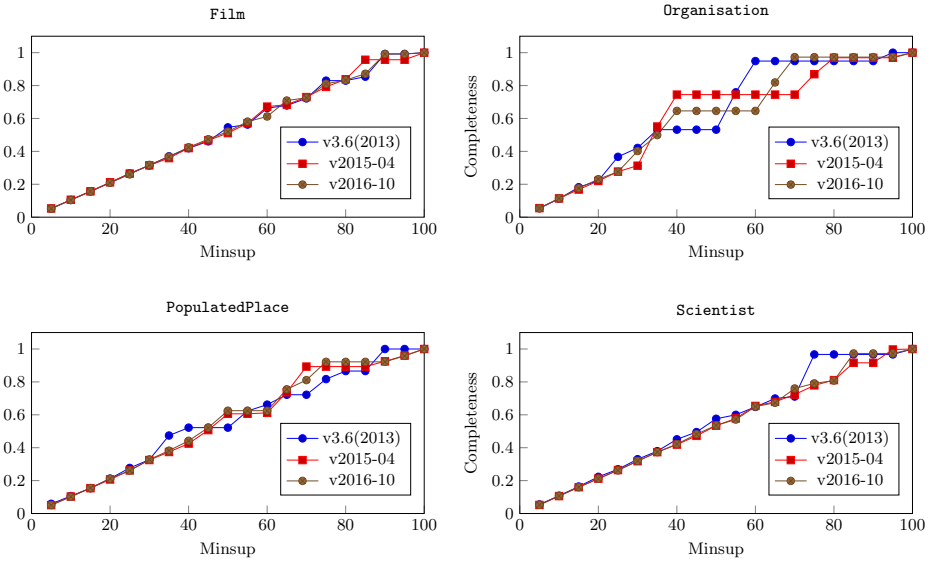

Fig. 1: Completeness of equivalent resources from DBpedia v3.6, v2015-04 and v2016-10

The results show that the diagrams of the classes *Films* and *PopulatedPlace* are roughly the same. Thus, we can conclude that for these classes, either the resources where almost not updated, either the updates preserve the completeness. For the class *Scientist* and *Organisation*, the results show that completeness seems better in 2013 version. This may be due to the addition of new widespread properties across those classes but not added for enough individuals, or maybe some individuals have lost some important properties while DBpedia evolving.

We then reproduced the same experiment but this time by taking all instances of a given class (not only instances belonging to the three datasets). The results of this new experiment are given in figure 2.

We observe for *Scientist* and for *Organisation* (except completeness for ξ between 30% and 50%), that completeness values are almost the same. As completeness was better in first expermiment in 2013 for those two classes, we may think that added individuals in 2015 and 2016 raised the completeness. However, for *Films* and especially for *PopulatedPlace* there is a clear difference in completeness values, and surprisingly 2013's and 2016's versions are very close. For the class *Films* the completeness values get lower in v2015-04 compared to the v3.6 version and the v2016-10. Either new *Film* instances lack of important properties when added, either updated *Film* instances have loss some of their important properties. Since in the first series of experiments, we see there are no differences for common instances for the three DBpedia versions, we can conclude that the first justification must be the right one. For the *PopulatedPlace*

class, as updated instances did not change the completeness, the result we can observe may be caused by instances added in 2015 that were more complete than those added in 2016, because the completeness went back down to the level of 2013 in 2016 after a rise in 2015.

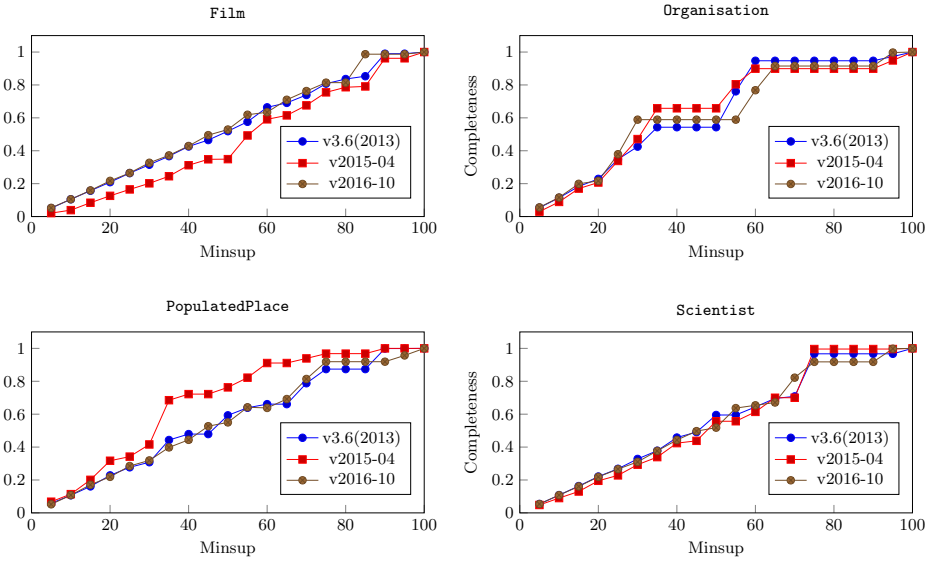

Fig. 2: Completeness of DBpedia v3.6, v2015-04 and v2016-10

6 Conclusion

This article is an exploratory study on the evolution of DBpedia completeness. We have first presented a completeness calculation approach. We then conducted a set of experiments upon three relatively distant versions of DBpedia, and we applied our proposed approach to four classes of DBpedia, any other classes could have been used instead. These experiments revealed that datasets completeness could increase or decrease due to changes made to exiting data or to the new added data. We also noticed that often this evolution does not benefit from the initial data cleaning as the set of properties continue evolving over time. Our approach could be helpful for data source providers to improve, or at least to keep a certain completeness of their datasets over different versions. It could be particularly useful for datasets constructed collaboratively, by imposing to contributors some rules when they update or add new resources. In the future, we plan to enrich our investigation with other data sources such as Yago, IMDB, etc. We would like to study the reasons why some categories improved their

completeness over time while other do the opposite. We also plan to improve the proposed approach by randomizing the selection of classes and increasing the number of those selected classes. We are currently working on an approach for Linked Data completeness improvement directed by the dataset content, its completeness results and *owl:sameAs* links.

References

1. Batini, C., Cappiello, C., Francalanci, C., Maurino, A.: Methodologies for data quality assessment and improvement. ACM Computing Surveys (CSUR) 41(3), 16 (2009)
2. Eastman, C.M., Jansen, B.J.: Coverage, relevance, and ranking: The impact of query operators on web search engine results. ACM Transactions on Information Systems (TOIS) 21(4), 383–411 (2003)
3. Grahne, G., Zhu, J.: Efficiently using prefix-trees in mining frequent itemsets. In: Proceedings of the ICDM 2003 Workshop on Frequent Itemset Mining Implementations, 19 December 2003, Melbourne, Florida, USA (2003)
4. Han, J., Pei, J., Yin, Y., Mao, R.: Mining frequent patterns without candidate generation: A frequent-pattern tree approach. Data Min. Knowl. Discov. 8(1), 53–87 (Jan 2004)
5. Hartig, O.: Trustworthiness of data on the web. In: Proceedings of the STI Berlin & CSW PhD Workshop. Citeseer (2008)
6. Herzig, D.M., Tran, T.: Heterogeneous web data search using relevance-based on the fly data integration. In: Proceedings of the 21st World Wide Web Conference 2012, WWW 2012, Lyon, France, April 16-20, 2012. pp. 141–150. ACM (2012)
7. Hogan, A., Harth, A., Passant, A., Decker, S., Polleres, A.: Weaving the pedantic web. In: Proceedings of the WWW2010 Workshop on Linked Data on the Web, LDOW 2010, Raleigh, USA, April 27, 2010 (2010)
8. Mendes, P.N., Mühleisen, H., Bizer, C.: Sieve: linked data quality assessment and fusion. In: Proceedings of the 2012 Joint EDBT/ICDT Workshops. pp. 116–123. ACM (2012)
9. Pipino, L.L., Lee, Y.W., Wang, R.Y.: Data quality assessment. Communications of the ACM 45(4), 211–218 (2002)
10. Schmachtenberg, M., Bizer, C., Paulheim, H.: Adoption of the linked data best practices in different topical domains. In: Semantic Web Conference (1). vol. 8796, pp. 245–260 (2014)
11. Wang, R.Y., Strong, D.M.: Beyond accuracy: What data quality means to data consumers. Journal of management information systems pp. 5–33 (1996)
12. Zaveri, A., Rula, A., Maurino, A., Pietrobon, R., Lehmann, J., Auer, S.: Quality assessment for linked data: A survey. Semantic Web 7(1), 63–93 (2016)